

# Centering ontologies in agent oriented software engineering processes

Ghassan Beydoun<sup>1</sup> · Graham Low<sup>2</sup>

Received: 6 March 2016 / Accepted: 14 September 2016 / Published online: 26 September 2016  
© The Author(s) 2016. This article is published with open access at Springerlink.com

**Abstract** A plethora of Multi Agent Systems (MAS) development methodologies exists and all compete for prominence. This paper advocates unification of best of breed activities from these methodologies and examines two existing approaches for unifying access to them. It proposes an alternative approach that focusses on the use of domain knowledge through ontologies as offering the best potential for unifying access to them. The reliance on ontologies will provide flexibility in the process and workproducts use within the methodology. The focus on domain knowledge will reduce the number of mandatory methodological tasks and at the same time create scope for reuse with respect to both system designs and components. The paper will further sketch and argue for a full software development lifecycle for MAS where ontologies expressing domain knowledge are the central artifacts.

**Keywords** Process elements · Design · Ontologies · Mutli agent systems · Distributed systems

## Introduction

Increasing interest in engineering a class of distributed intelligent systems, Multi Agent Systems, has led to growing attention to higher level software engineering reuse issues.

✉ Ghassan Beydoun  
ghassan.beydoun@uts.edu.au  
  
Graham Low  
g.low@unsw.edu.au

<sup>1</sup> Faculty of Engineering and IT, University of Technology Sydney, City Campus, Sydney, NSW, Australia

<sup>2</sup> School of Information Systems, University of New South Wales, Sydney, NSW 2052, Australia

This includes reuse of models, reuse of project development know-how, as well as that of project management knowledge [9]. As in any software development project, reuse should be of concern also to Multi Agent Systems (MAS) development. In other words, reuse of MAS models and MAS developers' skills should be facilitated. We argue in this paper that both dimensions of reuse can be facilitated with the use of ontologies as central constructs to drive the whole of the Software Development Life Cycle (SDLC). This paper promotes ontology-based MAS development. Substantial integration between ontologies and software engineering has been achieved e.g. in ODE of [15] and Onto [19]. This paper is part of an ongoing effort to (i) place ontologies at the centre of the software development lifecycle (SDLC) for MASs, (ii) enhance the reuse of MAS work-products and, (iii) unify agent-based software engineering knowledge.

The unique characteristics of MAS have rendered most standard systems development methodologies inapplicable, giving rise to the development of Agent Oriented Software Engineering (AOSE) methodologies. It should be said that, however, the iterative nature of the process remains common to all those methodologies. This typical requirement for the development of any knowledge based information systems is maintained. In other words, all the new methodologies continue to iteratively observe the common development phases of: *requirement definition, analysis, design and implementation* [27]. However, systematic review of those methodologies [1,29,30] have revealed that most still do not consider longer term issues associated with the usage of a MAS. They do not consider the reuse of the system by incorporating extensibility. They do not consider longevity of the system by facilitating maintenance and/or interoperability. Furthermore, they not protect the investment by ensuring that the workproducts of the development project are sufficiently reusable. This paper outlines a research agenda and a method-

ology creation path towards resolution of those long term issues. Through the use of ontologies during the software development lifecycle and at the same time the accommodation of iterative process models, the paper sketches a skeleton of an ontology-based process.

Towards this, Beydoun et al. [5–8] proposed development of agents in a way to decouple domain knowledge from problem solving knowledge. This is to enhance reuse in agent systems development. The domain knowledge would be sourced via the use of reusable *domain-dependent ontologies*. The problem solving/domain processes knowledge would be sourced via the use of reusable *problem-solving methods* (PSMs). E.g. these correspond to business process units in a set of similar application domains. They are in essence high-level structures that describe a reasoning process employed to solve similar problems encountered across different domains [3, 10, 31]. Both components, domain ontologies and problem solving methods would then be reused through deployment of a library of modular components. This would assist the domain-independent development of agent-oriented systems, reducing development costs and speeding up the development process. The work described in this paper continues on from [7, 12]. The roles of the reusable components interfaces are analysed in the design of a library of reusable components specifically oriented towards agent-oriented software engineering. During the design phase and selection of reusable components, agents are interactions are analysed to provide pointers to identify of the most suitable reusable component. As such, *interaction-dependencies specifications* extend the library of components to guide developers towards the design of solutions where is necessary for problem-solving, with interaction arising both in cooperation and negotiation of agents.

The rest of the paper is organised as follows: Section “An ontology-centric MAS software engineering scheme” provides the conceptual underpinning for placing ontologies at the heart of the Software Development Lifecycle (SDLC) in developing multi agent systems. Section “Ontology-based methodological development of MAS” develops this into a sketch of an iterative ontology-based methodology and describes how this can be adapted to a given specific context. Finally, Section “Conclusion” with a summary and discussion of future work.

## An ontology-centric MAS software engineering scheme

Conceptual modelling within knowledge engineering seeks to define the key concepts and relationships that constitute knowledge in any given context [22, 28]. The expression of such models in the form of ontologies provides the basis for re-use of these concepts and relationships in the form

of software in conjunction with complementary organizational processes [17]. The complexity of knowledge makes the development of a single and complete corpus impractical [28] which has, in turn, led to the proliferation of ontologies to account for differing contexts. Even within the same development context, it is common that a number of ontologies are relevant. The availability of multiple ontologies is considered here as a potentially useful resource for MAS development. To exploit this resource, MAS methodologies need to incorporate ontology oriented analysis and design tasks. Those tasks require a varying degree of abstractions and provide workproducts of varying degree of formality as reflected by the SDLC that they support. In essence, the work here supports a particular architecture design of distributed system, MAS. Within this architecture, ontologies in combination with the proposed tasks form a set of software patterns that facilitate reuse and development of a MAS. Various abstractions within the same ontology or multiple ontologies can support the analysis and design tasks within the various phases of the development of a MAS. The tasks that can be supported by ontologies will be identified in this paper.

A MAS is composed of a heterogeneous collection of agents with distinct knowledge-bases and capabilities. Coordination and cooperation between agents facilitate the achievement of collective goals which cannot be otherwise achieved by a single agent working in isolation [33]. Only a small number of existing MAS methodologies include ontologies in their work products and processes. This support is generally confined to the early phases of the development, to support the internal consistency of models during the *analysis* phase. Although, the agent methodology MOBMAS suggests that ontologies could be used to verify the structure of models, to support interoperability and reuse [29, 30], no explicit process towards that end is defined. Towards such a process, two significant innovations are required in support of Agent Oriented SE (i) an ontology-based methodological framework that can be used to build new ontology-centric AOSE methodologies from scratch, and (ii) a repository of add-on methodological elements that can be added to an existing AOSE methodology.

With all the reuse advantages stipulated for such a process, it would also address a broader concern of reuse- that of reusing SE knowledge itself. As earlier discussed, there is a growing realization that some innovative form of consolidation is needed. To produce a more effective methodological approach from the existing body of agent-oriented software engineering knowledge, it is important to first weight the pros and cons of the roadmap forward.

Two key issues are crucial to consider: (i) how easy it is for software developers to actually apply the unified outcome (ii) how feasible is the merging approach in the first place.

Two approaches that have been previously used to are now discussed from the these two perspectives.

The **ad-hoc approach** effectively merges existing methodologies one at a time, with an arbitrary methodology as a starting point, and without guidance on attaching methodologies. This approach is *not feasible*. An ad hoc approach can lead to repetition and to an unnecessarily large and cumbersome methodology. This should not come as a surprise as any given methodology targets a specific concern. We find that they greatly overlap in modelling and steps. An excessively large methodology would lead software developers subsequently struggle to deal with and they would most likely abandon the corresponding development (Cossentino et al. 2004).

The second is a **metamodeling approach** relying on a formally unifying formal language (a metamodel) to express various methodology fragments from different sources (perhaps using a metamodel such as FAML (FAME<sup>1</sup> Agent-oriented Modelling Language) as described in [7]. This approach requires a collection of versatile methods to create a repository of method fragments. A project manager subsequently uses the unifying language and decides the concern and the flavour required. This approach places *excessive* burden to develop the repository and on the project manager. Despite avoiding inconsistencies or repetitions between methodologies, only selected subset of rewritten components of methodologies can be integrated at any one time, and this is very time-consuming. Furthermore, the emerging area of AOSE, the development experience is limited and the project criteria of selection may not be known a priori.

The discussions of Sections “Introduction” and “An ontology-centric MAS software engineering scheme” advocate the support of libraries of components to support a unification end. But there is a need to balance the trade-off between reproducing the method fragments required by the unified language (in the metamodeling approach) and the cumbersome outcome of the ad hoc approach. This paper presents an alternative, and potentially complementary, approach using the domain ontology as a modelling artefact. Ontology techniques developed here can be used to enhance the metamodeling approach.

To avoid the cumbersome re-writing of existing methodologies using a common formal language (metamodel), a less formal feature-identification step is proposed. The approach requires much less effort and it is realizable without requiring any collaboration of the creators of the ontologies. This approach can also rid developers of the highly specialised and difficult task of the merging of methodology components on a per project basis. This is because the approach relies on using explicit ontologies as a focal point during the devel-

opment, and in this way facilitates combining features from different AOSE methodologies, using ontologies as a means for semantic mappings to convert software work products to suit various development steps. Substantially supporting integration of processes and products, this can productively support its inter-operation with other systems, and can be considered as a significantly beneficial benefit.

Using off-the-shelf domain ontologies as a starting point of the system development phase will become the focus of our efforts on the applied use of ontologies in agent software engineering. In addressing interoperability and work product reuse, the issue is not simply that an ontology-based AOSE methodology should be complete and consistent and produce systems that can easily be evolved to new contexts. The deeper issue is that it should have a highly developed maintenance phase and guide developers in reusing existing systems and components previously developed. Reuse is a central aim of using ontologies in general. Ontologies are in essence reusable encapsulation of knowledge [11,32,34]. In deploying them in MAS development processes, potentially lowers the long term cost of deployment of agent-based systems.

Two significant contributions to the state-of-the-art in agent based software engineering are identified as follows:

**First**, designers will have a tested and verified framework to handle interoperability issues at design time. This facilitated by creating the multi agent system from loosely coupled components connected through ontological mappings. Being inherently flexible with its actual design and architecture reusable across applications and in different settings, this ontology-driven approach is expected to be highly effective.

**Second**, ontological commitments related to the design of the multi agent system will be explicitly made during its actual design and development. In exploring the currently overlooked ontology-related interactions between the analysis and design phases of software development for MAS, iterative verification during the design and development of the system becomes also possible. A validation approach to complement this ontology-based view was recently developed in [20] to precisely use ontologies during the creation of the MAS requirement models. With respect to improving the process of creating intermediate software work products, using an ontology-based development methodology is impossible without the appropriate process elements that can harness domain knowledge within ontologies. In other words, the presence of problem solving methods is couple with partial usage of any ontology for any single agent. Thus, process elements are required to identify the “bits” within an ontology that are required at any single time. That is why the next section focuses on identifying the process elements required to create an ontology-driven methodology.

<sup>1</sup> FAME is the project name under which FAML has been developed.

## Ontology-based methodological development of MAS

An “ontology driven information system” is a system in which the ontology is an integral component of the system. For “an ontology driven” multi-agent system, an ontology is an integral part of each agent in the system and the system itself. In other words, we assume an ontology-based view of agents, in which every agent is assumed to have a local ontology reflecting their knowledge base and their point of view the overall systems requirements. Under this view, an agent within a multi agent system is an individual problem solver that requires knowledge of its operating domain and problem solving knowledge to enable appropriate communication with other agents and appropriate manipulation of its data environment. In this view, we categorize the knowledge of an agent into domain knowledge and into problem solving knowledge. The problem solving knowledge equates to techniques necessary to use an ontology and to turn it into a working system. Specific techniques for different kinds of problems are necessary to build relatively complete and competent systems. We pursue ontology-based processes to leverage the use of ontologies. This will also lead to identification of identifying and reusing concomitant problem-solving methods (PSM). Thus, supporting ontology based development also requires providing guidance to developers to identify PSM for individual agents and integrating these appropriately within a MAS, operationalising individual agents in concordance with requirement analysis models.

One major appeal of ontology-based development activities is that the reliance on analysis and design is reduced where the re-engineering of ontology is coupled with a suitable choice of problem solving methods [3]. Unlike many knowledge based development methodologies, this lacks the support of MAS development community. As a consequence, we generalize the early efforts of knowledge based systems researchers e.g. [21,24] and view a knowledge-based system (KBS) as comprising problem solving methods and a suitable sets of ontologies.

In our ontology-centric view of agents, each agent has its own view on the domain ontology, and the individual agents work to achieve their set of goals, determined by the systems requirements analysis, with the coordination between agents playing a key role in the system. The system overall possesses diverse knowledge and problem-solving capabilities embodied in the constituent agents. A MAS architecture as such addresses an answer to a number of well-known shortcomings of general problem solving methods [26]: incomplete knowledge requirement specification, incomplete problem solving requirement and limited computational resources. Those problems, however, may still apply at an individual agent level and their implications to the pursuit of an ontol-

ogy centric development process are instructive indeed. To individuals agents within a MAS they translate as follows:

- (i) some agent views of the domain may be incomplete; that is, their individual ontologies may be incomplete.
- (ii) individual problem solving knowledge for some individual agents may be insufficient for their own local goals and required behavior within the multi agent system.
- (iii) The multi agent system may still have limited execution resources despite the fact that multiple and parallel processing may be easier to achieve.

In other words, the above three issues can complicate the usage of ontologies at the individual agent level. A new problem/issue should also be thrown in that mix; agents need to share their results and communicate, whilst at the same time they have different ontologies. In other words, a common terminology/ontology of sorts is required to stitch the behavior of all the agents. Without careful design, this can be made even more difficult if the output of the local problem solvers described of individual agents operate at varying levels of abstraction. This may be simple to resolve if the outputs are complementary, but more challenging to resolve if they have varying degrees of prescription to the domain. Various degrees of adjustment to suit the domain would then be required. In other words, the level of specificity they exhibit to a given domain may greatly vary. For instance in the Belief-Desire-Intention (BDI) [23,25] architecture of agents, the use of knowledge (in encoding beliefs) was never intended to simultaneously accommodate various knowledge abstractions for the same domain. Considering those differences in abstractions between various ontologies when developing MAS, three ontology centric constraints/tasks are required in the development process. These are as follows.

1. Ontology mappings are required to allow individual agents to interact and to share the results of their local problem solving capabilities. In other words, a common domain conceptualization is required.
2. We need to ensure that the problem solving methods of individual agents have access sufficient domain knowledge to undertake its roles within the system. i.e. the verification of individual agents knowledge requirements against allocated ontologies is required at design time.
3. Knowledge extensibility is required at the agent level to accommodate any new ontological units added to the system about the domain. The individual agents view of the domain ontology is not necessarily complete. Despite the fact that this can create inconsistencies, it may be useful for monitoring purposes. However, a structured and understood knowledge representation is required



to resolve existing inconsistencies. For example, using incremental approaches based on interactions between an expert and a data stream input can be used [4, 10].

In the next section, we elaborate the above four analysis and design activities required to create an ontology-based development process.

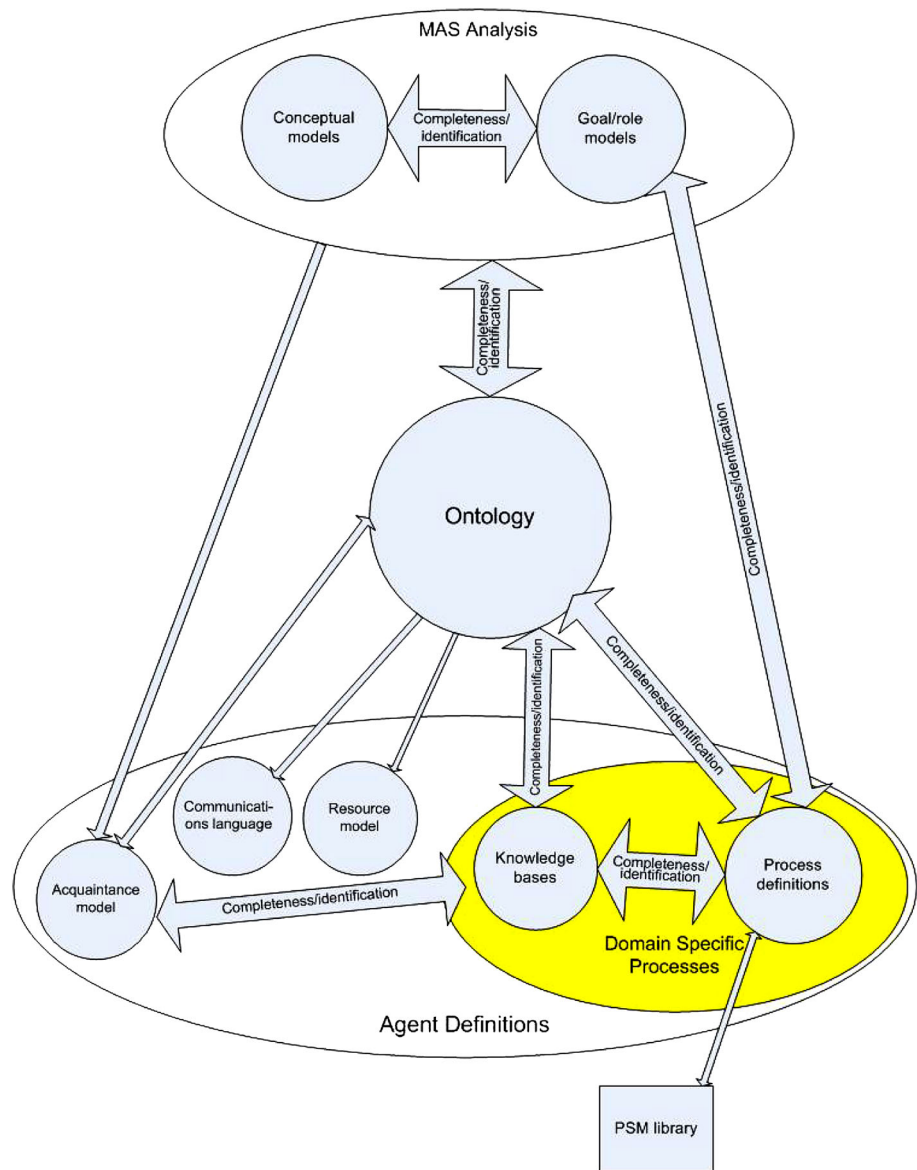
### Proposed ontology-driven methodology

We present the outline of our proposed ontology driven-methodology in Fig. 1. We have made the following two assumptions in developing our approach:

1. The choice of a problem solving mechanism can be made independently of domain analysis.
2. A domain ontology describing domain concepts and their relationships is available. For instance, certain accounting practices being the same across many domains and industries. Such practices, if well documented and prescribed, can provide processes for problem solving methods and can be adapted using a domain ontology.

Indeed, the second assumption is not highly restrictive, as currently there are many repositories for domain ontologies. If required, this assumption can be weakened further by including a domain analysis phase first. This could be a first of developing the system to identify concepts and their relationships (e.g. as proposed in [13]). Some industries, e.g.

**Fig. 1** A proposed interaction among ontology, models, communication languages and domain specific processes



banking and finance, may also be inclined to provide such ontologies. Given a domain ontology, the four process tasks discussed in the previous section and constraints earlier identified, we sketch features of the analysis and design activities for an ontology-based development process.

An inter-play between the various facets of reuse facilitated by the ontology-based approach is expected. For example, the role of the ontology in providing interoperability at run-time requires careful consideration of run-time temporal requirements. An ontology's role in reasoning at run-time is based on fulfilling the problem solving requirements of agents at design time. This requires scoping the domain analysis for each individual agent at design time. However, a preliminary step is identifying the relevant domain ontology (ies) for the whole development lifecycle of the MAS. In recent work [9], we developed a retrieval mechanism based on requirement models identified using goal analysis. In other words, given an ontologies repository it becomes a matter of providing the goal models to identify the required ontologies. In some domains, it may be possible to source multiple domain ontologies and the analyst can be given an opportunity to select a more appropriate ontology. An ontology evaluation mechanism that uses the structure of the ontology can then be integrated. An example of such a mechanism is detailed in [11].

Goal models are the usual way used to express requirements models for agent based systems (e.g. [16,33]). Supported by the domain ontology, an early requirement phase can generate a high level description of system goals (and roles) and a high level conceptual description of the system (shown as *conceptual models* in Fig. 1). This description includes contextual description of the environment of the system (typical agent oriented models that would be included here may include organizational models and environmental models).

Hence, we envisage the ontology-based development process to incorporate support for goals models co-evolution via the iterative nature of the early requirements identification and validation. As Fig. 1 shows, this can occur between the domain ontology model and the requirements models, and there can be other iterative processes in deriving the system goals and roles from the requirements models (e.g. using other available conceptual models as shown in Fig. 1).

The requirement models can also provide for further validation of the domain ontology, with any inconsistencies noted causing another iteration of requirements models identification. With each iteration, the ontology itself may also be improved. Any incompleteness of the domain ontology can trigger further domain expert advice by the requirement engineers. The role models and goal models for instance are further refined to provide a clear association between agent roles and lower level goals to permit associating problem

solving capabilities (using PSM libraries) and system goals in the early stages of the system design.

Chosen problem solving capabilities for different agents in a given multi agent system do not necessarily have the same required degree of domain dependence. That is, for a particular chosen agent capability, the domain ontology required may need to be adapted. For this purpose, the domain ontology is again construed as the first reference point. Ontology mapping between portions of the domain ontology and the local agent's knowledge is required to ensure that all agent capabilities have their knowledge requirement available to their reasoning format. For inter agents communication, a global communication language derived directly from the domain ontology would suffice [14]. The communication language, as shown in Fig. 1, enables messages to be exchanged between agents and it is a low level design construct rather than an analysis construct (as is the case with the domain ontology).

In addition, an agent's problem solving capability cannot be assumed to be sufficiently powerful to respond to all events it encounters during its lifetime. Current practices often assume that functional goal analysis is sufficient to specify the knowledge requirement for agents [16], and any deficiencies in its later problem-solving capacity are assumed to be offset by cooperation. However, in our view, without consideration of its actual problem solving capacity (and others available within the system), there is no guarantee that this cooperation would ultimately work. This suggests that iteration between the problem solving mechanism design and the requirements analysis is required to ensure that the chosen problem solver for a given agent is capable of meeting its specified goals.

In summary, as shown in Fig. 1, The Agent Definition shows three models (acquaintance, communications and resource) that are updated as the domain ontology is updated. The acquaintance model stores information about collaborating agents and would be developed to support the agent requirements in MAS analysis. Changes in the acquaintance model resulting from changes in MAS requirements would need to be reflected in the domain ontology as indicated by the two-way interaction. In addition specific local agent The Resource Model is developed from the domain ontology to show which ontologies conceptualize each resource used by the multi-agent system, while the communications language would need to be appropriate for the domain ontology.

## Conclusion

The paper presented an agent oriented software engineering approach that facilitates combining features from different agent oriented methodologies. It employs ontologies as

a means for semantic mappings and effectively converts software work products to outfit a variety of developmental steps. Substantially supporting the integration of processes and products, the proposed approach can also effectively support its inter-operation with the interacting systems.

The approach presented supports the selection of a MAS architecture. In the broader context of software engineering, architectures facilitate the design of the system and the transition from requirement models to design models and can reduce the cost of development. The choice of MAS architecture changes the requirement analysis process and how the requirement models are first synthesized. Whilst MAS architectures can also contribute to cost management of a project [2], they are often pursued as a complexity management/problem solving tool which may in some cases allow tackling new problems [18]. Implementing a MAS architecture can benefit from a specific requirement analysis approaches to transform the requirements into appropriate models that can then be used to derive a MAS. In this paper, we advocate the use of ontologies to facilitate this transformation of the requirement models into design and implementation (models).

Based on a sketched methodology provided, we also showed that our approach can be enhanced in at least three different directions. First, localised agent ontologies (to support their capabilities) and a global agents communication ontology can be derived to support the system's development. Second, for further efficiency, a facilitating ontology mapping can be enforced between localised ontologies and the communication ontology. Based on the corresponding domain ontology, the same adaptation between the reasoning and domain ontology can be used to map the result of reasoning back to a common communication ontology. Further schematic schemes can be provided to further discuss the directions in which the preferred approach can be enhanced.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

- Argente E et al (2011) Modelling with agents. In: Gleizes M-P, Gomez-Sanz J (eds) Agent-oriented software engineering X. Springer, Berlin, pp 157–168
- Benfield SS, Hendrickson J, Galanti D (2006) Making a strong business case for multiagent technology. In: Proceedings of the 5th international joint conference on autonomous agents and multiagent systems, Hakodate, Japan
- Benjamins VR, Plaza E, Motta E, Fensel D, Studer R, Wielinga B, Schreiber G, Zdrahal Z (1998) IBROW3: an intelligent brokering service for knowledge-component reuse on the world wide web. Banff Knowledge Acquisition Workshop (KAW98). Canada
- Beydoun G, Hoffmann A (2013) Dynamic evaluation of the development process of knowledge-based information systems. *Knowl Infor Syst* 35(1):233–247
- Beydoun G, Tran N, Low G, Henderson-Sellers B (2006a) Foundations of ontology-based methodologies for multi-agent systems. In: Kolp M, Bresciani P, Henderson-Sellers B, Winikoff M (eds), *Procs. AOIS2005 LNAI 3529*, Springer, Berlin, pp 111–123
- Beydoun G, Gonzalez-Perez C, Henderson-Sellers B, Low GC (2006b) Developing and evaluating a generic metamodel for MAS work products. In: Garcia A et al. (ed.) *Software engineering for multi-agent systems IV: research issues and practical applications*, vol. LNCS 3914, Springer, Berlin, pp 126–142
- Beydoun G, Krishna AK, Ghose A, Low GC (2009) Towards ontology-based MAS methodologies: ontology based early requirements. In: Barry C, Lang M, Wojtkowski W, Wojtkowski G, Wrycza S, Zupancic J (eds) *The inter-networked world: ISD theory, practice, and education*. Springer, New York, pp 923–935
- Beydoun G, Low G, Henderson-Sellers B, Mouraditis H, Sanz JJG, Pavon J, Gonzales-Perez C (2009b) FAML: a generic metamodel for MAS development. *IEEE Trans Softw Eng* 35(6):841–863
- Beydoun G, Low G, García-Sánchez F, Valencia-García R, Martínez-Béjar R (2014) Identification of ontologies to support information systems development, information systems. Elsevier, Amsterdam 46:45–60
- Beydoun G, Hoffmann A (1998) Simultaneous modelling and knowledge acquisition using NRDR. In: 5th Pacific rim conference on artificial intelligence (PRICA198), Springer, Singapore
- Beydoun G, Lopez-Lorca A, Garcia-Sanchez F, Martinez-Béjar R (2011) How do we measure and improve the quality of a hierarchical ontology? *J Syst Softw* 84(12):2363–2373
- Brown R, Beydoun G, Low G, Tibben W, Zamani R, García-Sánchez F, Martinez-Bejar R (2016) Computationally efficient ontology selection in software requirement planning. *Inf Syst Front* Springer 18(2):349–358
- Cordi V, Mascardi V, Martelli M, Sterling L (2004) Developing an ontology for the retrieval of XML documents: a comparative evaluation of existing methodologies. In: *Proceedings of agent oriented information systems workshop (AOIS2004)*, Riga, Latvia
- Esteve M, De La Cruz D, Sierra C (2002) ISLANDER: an electronic institutions editor. In: *International conference on autonomous agents and multiagent systems (AAMAS02)*, Italy, pp 1045–1052
- Falbo RA, Arantes DO, Natali ACC (2004) Integrating knowledge management and groupware in a software development environment. In: *5th International conference on practical aspects of knowledge management*, Vienna, Austria, Springer, Berlin
- Giunchiglia F, Mylopoulos J, Perini A (2003) The tropos software development methodology: processes, models and diagrams. In: Giunchiglia F, Odell J, Weiß G (eds) *Agent-oriented software engineering III: 3rd international workshop, AOSE 2002*. Springer, New York, pp 162–173
- Guizzardi G (2007) On ontologies, conceptualizations, modelling languages and (meta) models. In: Vasilecas O, Elder J, Caplinskas A (eds) *Databases and information systems IV*. IOS Press, Amsterdam, pp 18–39
- Lamparter S, Becher S, Fischer J (2010) An agent-based market platform for smart grids. In: *Autonomous agents and multi agent systems conference (AAMAS2010)*, Toronto
- Leppänen A (2007) A context-based enterprise ontology. *Lecture notes in computer science*, 4439, Springer, New York, pp 273–286

20. Lopez-Lorca A, Beydoun G, Valencia-Garcia R, Martinez-Bejar R (2016) Supporting agent oriented requirement analysis with ontologies. *Int J Hum Comput Stud* 87:20–37
21. Motta E, Zdrahal Z (1996) Parametric design problem solving. In: 10th Banff knowledge acquisition for knowledge based system workshop, Canada
22. Othman SH, Beydoun G, Sugumaran V (2014) Development and validation of a disaster management metamodel. *Inf Process Manag* 50(2):235–271
23. Padgham L, Lambrix P (2000) Agent capabilities: extending BDI theory. 17th National conference on artificial intelligence (AAAI-2000). Austin, Texas USA, pp 68–73
24. Puppe F (1993) Systematic introduction to expert systems: knowledge representation and problem-solving methods. Springer, Berlin
25. Rao AS, Georgeff MP (1995) BDI agents: from theory to practice. In: Proceedings of the 1st international conference on multi-agent systems, pp 312–319. San Francisco, CA
26. Russell S, Norvig P (2009) Artificial intelligence, a modern approach, the intelligent agent book, 3rd edn. Prentice Hall, Berkley
27. Sadrei E, Aurum A, Beydoun G, Paech B (2007) A field study of the requirements engineering practice in Australian software industry. *Requirements Eng* 12(3):145–162
28. Shreiber G, Akkermans H, Anjewierden A, Hoog R, Shadbolt N, De Velde WV, Wielinga B (2001) Knowledge engineering and management: the CommonKADS methodology. The MIT Press, London
29. Tran QNN, Low GC (2006) A methodological framework for ontology centric agent oriented software engineering. *Int J Comput Syst Sci Eng* 21:117–132
30. Tran QNN, Beydoun G, Low G (2007) Design of a peer-to-peer information sharing MAS using MOBMAS (ontology-centric agent oriented methodology. *Advances in information systems development*. Springer, New York, pp 63–76
31. Trokanas N, Cecelja F (2016) Ontology evaluation for reuse in the domain of process systems engineering. *Comput Chem Eng* 85:177–187
32. Wang H, Wang S (2016) Application of ontology modularization to human-web interface design for knowledge sharing. *Exp Syst Appl* 46:122–128
33. Wooldridge M (2002) An introduction to multi agent systems. Wiley, Chichester
34. Xu D, Wijesooriya C, Wang X-G, Beydoun G (2011) Outbound logistics exception monitoring: a multi-perspective ontologies' approach with intelligent agents. *Exp Syst Appl* 38(11):13604–13611